SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

SYSTEM, METHOD AND COMPUTER PRODUCT FOR INCREMENTAL IMPROVEMENT OF ALGORITHM PERFORMANCE DURING ALGORITHM DEVELOPMENT

Federal Research Statement

The US Government may have certain rights in this disclosure pursuant to Government Contract Number MDA 972-98-3-0002 awarded by the Defense Advanced Research Projects Agency (DARPA).

Background of Invention

[0001] This disclosure relates generally to algorithm development and more particularly to assessing the performance of an algorithm at different phases of development and assessing the impact that input variations have on the algorithm.

[0002]

Generally, an algorithm undergoes several validations during the course of its development. Validation is the systematic evaluation of the algorithm performance. The evaluation includes performing reviews and tests to ensure that functional requirements are complete and testable. Generally, a review is a form of inspection that includes reviewing documentation to make sure that it supports operation, maintenance and future enhancements. In addition, the review includes determining whether established requirements, design concepts and specification have been met. Generally, a test includes operating the algorithm with real or simulated inputs to demonstrate that it satisfies the functional requirements. Typically, during a test, various test cases having selected input data and parameters are used as inputs to the algorithm. Outputs and reactions of the algorithms are then noted. Test failures are used to identify the specific differences between expected and actual results.

[0003] There are several drawbacks associated with the above-noted process for validating algorithms. Most notably, when the choice of algorithms to be implemented is not known a priori, for example because the algorithms are tasked with solving a new problem that has no known benchmark solution, the impact of newly developed parts of the algorithm on solving the stated problem cannot easily be judged. In addition, if the problem domain is poorly known, algorithmic solutions are carried without the benefit of knowing exactly the impact on the unknown parts of the problem domain. This predicament is compounded by several factors such as increasing complexity of the problem domain, multiple boundary conditions, and permutation of possible cases.

[0004] In order to overcome the above drawbacks, there is a need for a methodology that allows a software developer to assess the performance of an algorithm at different phases of development and assess the impact that input variations will have on the algorithm.

Summary of Invention

[0005] In one aspect of this disclosure, there is a system, method and computer readable medium that stores instructions for instructing a computer system, to assess the performance of an algorithm during development. In this embodiment, a design of experiments component establishes an acceptable number of experiments for analyzing the algorithm. An experiment performance component runs the established number of experiments for the algorithm. A simulation component simulates the behavior of the algorithm using results from the experiment performance component.

[0006]

In another aspect of this disclosure, there is a system, method and computer readable medium that stores instructions for instructing a computer system, to assess the performance of an algorithm during development. In this embodiment, a design of experiments component establishes an acceptable number of experiments for analyzing the algorithm. An experiment performance component runs the established number of experiments for the algorithm. A simulation component simulates the behavior of the algorithm using results from the experiment performance component. A simulation performance component evaluates the performance of the simulation for the algorithm.

[0007] In a third aspect of this disclosure, there is a system, method and computer readable medium that stores instructions for instructing a computer system, to assess the performance of an algorithm during development. In this embodiment, a design of experiments component establishes an acceptable number of experiments for analyzing the algorithm. An experiment performance component runs the established number of experiments for the algorithm and uses a performance metric to evaluate the results of the experiments. A Monte Carlo simulation component simulates the behavior of the algorithm using results from the experiment performance component with a Monte Carlo simulation. A simulation performance component evaluates the performance of the Monte Carlo simulation for the algorithm.

[0008] In still another aspect of this disclosure, there is a system, method and computer readable medium that stores instructions for instructing a computer system, to assess the performance of an algorithm during development. In this embodiment, a design of experiments component establishes an acceptable number of experiments for analyzing the algorithm. An experiment performance component runs the established number of-experiments for the algorithm. A performance metric component evaluates the results of the experiments run for the algorithm. An algorithm adjustment component adjusts logic or parameters of the algorithm for unacceptable results.

[0009] In a fifth aspect of this disclosure, there is a system, method and computer readable medium that stores instructions for instructing a computer system, to assess the performance of an algorithm during development. In this embodiment, a Monte Carlo simulation component simulates the behavior of the algorithm with a Monte Carlo simulation, wherein the Monte Carlo simulation component uses at least one confusion matrix to simulate the behavior of the algorithm. A simulation performance component evaluates the performance of the Monte Carlo simulation for the algorithm. An algorithm adjustment component adjusts logic or parameters of the algorithm for unacceptable results.

Brief Description of Drawings

[0010] Fig. 1 shows a schematic diagram of a general-purpose computer system in which a system for assessing the performance of an algorithm operates;

[0011] Fig. 2 shows a top-level component architecture diagram of an algorithm performance assessment system that operates on the computer system shown in Fig. 1;

[0012] Fig. 3 shows an architectural diagram of a system that implements the algorithm performance assessment system shown in Fig. 2;

[0013] Fig. 4 shows a flow chart describing actions performed by the algorithm performance assessment system shown in Fig. 2 as it operates within the system shown in Fig. 3;

[0014] Fig. 5 shows an example of a possible application for the algorithm performance assessment system shown in Fig. 2;

[0015] Fig. 6 shows a fusion algorithm process map for the information fusion tool shown in Fig. 5; and

[0016] Fig. 7 illustrates an example of simulating a diagnostic tool output for the information fusion tool shown in Fig. 5 with the algorithm performance assessment system shown in Fig. 2.

Detailed Description

[0017] This disclosure describes a system, method and computer product for assessing the performance of an algorithm during its development. Fig. 1 shows a schematic diagram of a general-purpose computer system 10 in which a system for assessing the performance of an algorithm operates. The computer system 10 generally comprises a processor 12, memory 14, input/output devices, and data pathways (e.g., buses) 16 connecting the processor, memory and input/output devices. The processor 12 accepts instructions and data from memory 14 and performs various calculations. The processor 12 includes an arithmetic logic unit (ALU) that performs arithmetic and logical operations and a control unit that extracts instructions from memory 14 and decodes and executes them, calling on the ALU when necessary. The memory 14 generally includes a random-access memory (RAM) and a read-only memory (ROM), however, there may be other types of memory such as programmable read-only memory (PROM), erasable programmable read-only memory (EPROM) and electrically erasable programmable read-only memory (EEPROM). Also, memory 14 preferably contains an operating system, which executes on the processor 12. The operating system performs basic tasks that include recognizing input, sending output to output devices, keeping track of files and directories and controlling various peripheral devices.

[0018] The input/output devices may comprise a keyboard 18 and a mouse 20 that enter data and instructions into the computer system 10. Also, a display 22 may be used to allow a user to see what the computer has accomplished. Other output devices may include a printer, plotter, synthesizer and speakers. A communication device 24 such as a telephone or cable modem or a network card such as an Ethernet adapter, local area network (LAN) adapter, integrated services digital network (ISDN) adapter, Digital Subscriber Line (DSL) adapter or wireless access card, enables the computer system 10 to access other computers and resources on a network such as a LAN, wireless LAN or wide area network (WAN). A mass storage device 26 may be used to allow the computer system 10 to permanently retain large amounts of data. The mass storage device may include all types of disk drives such as floppy disks, hard disks and optical disks, as well as tape drives that can read and write data onto a tape that could include digital audio tapes (DAT), digital linear tapes (DLT), or other magnetically coded media. The above-described computer system 10 can take the form of a hand-held digital computer, personal digital assistant computer, notebook computer, personal computer, workstation, mini-computer, mainframe computer or supercomputer.

[0019] Fig. 2 shows a top-level component architecture diagram of an algorithm performance assessment system 28 that operates on the computer system 10 shown in Fig. 1. The algorithm performance assessment system comprises a Design of Experiments (DoE) component 30 that establishes an acceptable number of experiments for analyzing an algorithm under development. The DoE is a well-known technique for mathematically selecting and analyzing experimental results to efficiently gather information relating a process response to selected independent variables. Generally, the DoE comprises steps such as defining a problem, establishing an objective, selecting response variable(s), selecting independent variable(s), choosing variable levels, selecting an experiment design, running the experiment and collecting data, analyzing the data and drawing conclusions. As one of ordinary skill in the art will recognize, these steps are dependent upon the particular application of the algorithm that is being developed.

[0020] In this disclosure, the DoE component 30 establishes the acceptable number of experiments based upon a selected subset of a design space defined by an expert. In this disclosure, the design space is the space described by the variables and parameters or logic that are input to the algorithm to be designed. The logic or parameters can be the output from systems that are part of the process before the algorithm to be designed. The acceptable number of experiments determined by the DoE component 30 generally depends on the level of accuracy desired and is further constrained by resource limitations such as financial resources or time resources. For example, using the DoE component 30 allows tailoring the number of solutions that the expert has to evaluate to the expert time available with a trade-off on accuracy. The DoE component 30 also selects an appropriate level of experiments for analyzing the algorithm. Generally, there are three levels of DoE; 1) full factorial, 2) fractional factorial or 3) screening. A full factorial experiment allows all combinations of all variables to be tested, while a fractional (e.g., half) factorial experiment allows many factors to be assessed simultaneously with reasonable economy of effort and a screening DoE isolates a "vital few" variables from a "trivial many" variables.

[0021] An experiment performance component 32 runs the number of experiments established for the algorithm by the DoE component 30. The experiment performance component 32 comprises a performance metric component 34 that evaluates the results of the experiments run for the algorithm using a performance metric. The performance metric contains evaluation criteria for comparing the results to a baseline algorithm. Examples of some performance metrics may include the solution specificity, sensitivity, robustness, execution time, false positives, false negatives, true positives, true negatives, false classifieds, etc. The baseline algorithm is an algorithm that would provide the best (by some metric) results if the algorithm undergoing assessment was not available. If no baseline algorithm exists, the first path through the newly developed algorithm (before improvements are being made) can be used as a baseline. The experiment performance component 32 also comprises an algorithm adjustment component 36 that adjusts the logic or parameters of the algorithm if the results determined by the performance metric component 34 are unacceptable. In particular, the algorithm adjustment component 36 changes the algorithm and logic or parameters if the results determined by the performance metric component 34 are unacceptable. In this case, the experiment performance component 32 will then run the experiments for the adjusted algorithm and logic or parameters.

[0022] A module insertion component 38 inserts an additional module into the algorithm if the results determined by the performance metric component 34 are acceptable. The experiment performance component 32 then runs the experiments for the algorithm including the inserted module. The module insertion component 38 will continue to add more modules into the algorithm until performance is satisfactory. Likewise, the experiment performance component 32 will run the experiments for the algorithm including the inserted modules. This iterative process enables a designer to assess the performance of the algorithm as each additional component is inserted.

[0023] A simulation component 40 simulates the behavior of the algorithm using the results obtained from the experiment performance component 32. The simulation component 40 uses a Monte Carlo simulation to simulate the behavior of the algorithm. A Monte Carlo simulation is a well-known simulation that randomly generates values for uncertain variables over and over to simulate an environment in which a population or study exists. The Monte Carlo simulation also identifies sources of variation, relationships between covariates and links inputs to outputs. In this disclosure, the Monte Carlo simulation component 40 simulates the behavior of the algorithm on a full design space. In addition, the Monte Carlo simulation component 40 comprises a plurality of confusion matrices. A confusion matrix contains information indicative of classifier performance. In particular, the confusion matrix is defined as an $n \times n$ matrix where n is the number of classes considered by the classifier. The confusion matrix is arranged such that each of the rows contain true results for a particular class. The columns contain the output of the classifier. The resulting matrix shows the correctly classified output on the diagonal as the true positives (TP) and true negatives (TN). If the first class is dedicated to be the normal class (or null fault), then the following definitions can be made: The first row except the first entry contains the false positives (FP). The first column except the first entry contains the false negatives (FN). The off-diagonal entries except the first column and the first row contain the falsely classified (FC) classifier results. The confusion matrix can be normalized by dividing each entry of a row by the sum of the entries of the respective row. This provides a convenient way to read off the performance metrics.

[0024] A simulation performance component 44 evaluates the performance of the simulation for the algorithm. In particular, the simulation performance component 44 determines if the results of the simulation are acceptable. The criteria for determining whether the simulation results are acceptable will vary according to the algorithm and the user performing the assessment of the algorithm. If the results are acceptable, then the algorithm assessment is over. The simulation performance component also comprises an algorithm adjustment component 46 that adjusts the logic or the parameters of the algorithm if the results are unacceptable. If the parameters of the algorithm are adjusted, then the simulation component 40 runs the simulation for the system including the portion of the adjusted algorithm. The simulation performance component 44 then evaluates the performance of this simulation. This process (i.e., adjusting logic or parameters and running simulation) continues until the simulation performance component 44 determines that the results are acceptable.

[0025] Fig. 3 shows an architectural diagram of a system 48 that implements the algorithm performance assessment system 28 shown in Fig. 2. In Fig. 3, a computing unit 50 allows a user to access the algorithm performance assessment system 28. The computing unit 50 can take the form of a hand-held digital computer, personal digital assistant computer, notebook computer, personal computer or workstation. The user uses standard output to a window or a web browser 52 such as Microsoft INTERNET EXPLORER, Netscape NAVIGATOR or Mosaic to locate and display the algorithm performance assessment system 28 on the computing unit 50. A communication network 54 such as an electronic or wireless network connects the computing unit 50 to the algorithm performance assessment system 28. In particular, the computing unit 50 may connect to the algorithm performance assessment system 28 through a private network such as an extranet or intranet or a global network such as a WAN (e.g., Internet). As shown in Fig. 3, the algorithm performance assessment system 28 is provided by a server 56, which comprises a web server that serves the algorithm performance assessment system 28.

[0026] If desired, the system 48 may have functionality that enables authentication and access control of users accessing the algorithm performance assessment system 28. Both authentication and access control can be handled at the web server level by the algorithm performance assessment system 28 itself, or by commercially available packages such as Netegrity SITEMINDER. Information to enable authentication and access control such as the user names, location, telephone number, organization, login identification, password, access privileges to certain resources, physical devices in the network, services available to physical devices, etc. can be retained in a database directory. The database directory can take the form of a lightweight directory access protocol (LDAP) database; however, other directory type databases with other types of schema may be used including relational databases, object-oriented databases, flat files, or other data management systems.

[0027] Fig. 4 shows a flow chart describing actions performed by the algorithm performance assessment system 28 shown in Fig. 2 as it operates within the system 48 shown in Fig. 3. The flow chart begins at block 58, where an expert determines a design space and selects a subset of the defined design space at 60 for the algorithm that is undergoing development. The DoE component 30 then performs the DoE at 62. As mentioned above, performing the DoE includes establishing an acceptable number of experiments and choosing an appropriate level of experiments for analyzing the algorithm. Next, the experiment performance component 32 runs the number of experiments established for the algorithm at 64. These experiments reflect specific cases of system input behavior which is then simulated accordingly. The experiment performance component 32 then evaluates the results of the experiments run for the algorithm at 66. In particular, it evaluates the contribution of the algorithm component under development. This includes using a baseline algorithm provided at 68 and a performance metric provided at 70 and comparing the results obtained at 64 to the baseline algorithm and performance metric.

[0028] If the results as determined at 72 are not acceptable, then the algorithm adjustment component 36 adjusts the algorithm logic or the parameters of the algorithm at 74. The experiment performance component 32 then runs the experiments for the adjusted algorithm and logic or parameters at 64. The algorithm performance assessment system 28 then repeats blocks 66-72 until the results are acceptable. Next, the algorithm performance assessment system 28 determines at 76 if there are more algorithm components that need to be developed. If so, then other components are added to the algorithm at 78. The algorithm performance assessment system 28 then repeats blocks 64-78 until there are no more components. The decision to develop more algorithm components is made in part based on the overall performance of the algorithm, availability of resources and any potentially identified performance improving measures.

[0029] Once it is determined that there are no more components to insert into the algorithm, then the simulation component 40 simulates the behavior of the system using the results obtained from the experiment performance component 32 at 80. As mentioned above, the simulation component 40 uses a Monte Carlo simulation to simulate the behavior of the input system. The Monte Carlo simulation uses at least one confusion matrix obtained from a plurality of matrices at 90. The confusion matrices are obtained from running experiments at 88 on process data. The Monte Carlo simulation component 40 then uses the confusion matrices to simulate the behavior of the input system. To that end, a z-score is computed, which is a statistical measure indicative of how many standard deviations away from the mean the particular value is located, for a given fault case. The z-score is computed by interpreting the associated entries in the confusion matrix as the area under the curve of a Gaussian distribution from which the z-score is obtained through iterative integration. Then, random values are generated based on the z-score which reflect the behavior shown in the confusion matrices. The simulation performance component 40 evaluates the performance of the simulation for the algorithm at 82. In particular, if the simulation performance component 40 determines that the results of the simulation are acceptable, then the algorithm assessment ends. Otherwise, the algorithm adjustment component 36 adjusts the logic or parameters of the algorithm at 84 and modifies the algorithm components at 86. If the parameters or logic or components of the algorithm are adjusted, then the simulation is repeated at 80. Blocks 82-86 are repeated until the results are acceptable.

[0030] The foregoing flow chart of this disclosure shows the functionality and operation of the algorithm performance assessment system 28. In this regard, each block represents a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that in some alternative implementations, the functions noted in the blocks may occur out of the order noted in the figures or, for example, may in fact be executed substantially concurrently or in the reverse order, depending upon the functionality involved. Furthermore, the functions can be implemented in programming languages such as Matlab, C, Visual Basic or JAVA; however, other languages can be used.

[0031] The above-described algorithm performance assessment system 28 comprises an ordered listing of executable instructions for implementing logical functions. The ordered listing can be embodied in any computer-readable medium for use by or in connection with a computer-based system that can retrieve the instructions and execute them. In the context of this application, the computer-readable medium can be any means that can contain, store, communicate, propagate, transmit or transport the instructions. The computer readable medium can be an electronic, a magnetic, an optical, an electromagnetic, or an infrared system, apparatus, or device. An illustrative, but non-exhaustive list of computer-readable mediums can include an electrical connection (electronic) having one or more wires, a portable computer diskette (magnetic), a random access memory (RAM) (magnetic), a read-only memory (ROM) (magnetic), an erasable programmable read-only memory (EPROM or Flash memory) (magnetic), an optical fiber (optical), and a portable compact disc read-only memory (CDROM) (optical).

[0032] Note that the computer readable medium may comprise paper or another suitable medium upon which the instructions are printed. For instance, the instructions can be electronically captured via optical scanning of the paper or other medium, then compiled, interpreted or otherwise processed in a suitable manner if necessary, and then stored in a computer memory.

[0033] The algorithm performance assessment system 28 of this disclosure is suitable for assessing the performance of a multitude of algorithms during development. One example illustrating an application of the algorithm performance assessment system 28 is with the development of an information fusion tool for aggregating outputs from different diagnostic tools. Diagnostic information fusion tools are useful to service providers and manufactures because of their capability to gather and combine results obtained from different diagnostic tools. Generally, the information fusion tools maximize the advantages of each diagnostic tool, while at the same time minimizing their disadvantages. US Patent Application Serial Number 09/704,476, filed on November 3, 2000, and entitled "Information Fusion Of Classifiers In Systems With Partial Redundant Information" provides more information on one type of information fusion tool used to diagnose aircraft engine gas path faults. A description of how the algorithm performance assessment system 28 assesses the algorithms associated with the development of this type of information fusion tool follows. Note that the algorithm performance assessment system 28 of this disclosure is not limited to algorithms associated with information fusion tools and can be used in a variety of applications where it is desirable to evaluate the performance of an algorithm during the design process.

[0034] Fig. 5 shows a schematic of a fusion method used to diagnose aircraft engine gas path faults. In Fig. 5, raw engine measurements are provided to the system via an input bus 94 which may be a 1553 bus, an ARINC bus, or other appropriate data transmission medium. Relevant information is sent to the appropriate diagnostic tool such as those depicted in diagnostic functions block 96, and to the appropriate secondary evidence block 98. Outputs from the tools in diagnostic and evidencing blocks 96 and 98 are provided to an information fusion tool 100 as lists of fault identifications and performance detection. The information fusion tool 100 aggregates the multiple fault lists into a single list. This aggregation of fault lists addresses issues such as partial overlap of output information, missing information, conflicting information, different performance levels of the tools inputting the information, and the timeliness of the diagnostic data. As can be seen, by Fig. 5, portions of the fusion tool operation will occur while the aircraft powered by the engines under observation are airborne 102, while other actions are taken when the engines are on the ground 104. The operations taking place on the ground provide further refinement of outputs by comparing fused fault lists to historic and confirmed lists in order to obtain a final fused output.

[0035] The secondary evidential information block 98 contains existing or known types of information of system operations for gas turbine engines. For example, the model-based engine trending estimator is an existing piece of software which monitors operation of an engine and makes predictive statements as to the expected lifetime of the engine's operation. A vibration analysis detects the amount of vibration within an engine to determine damage which may be caused by the existence of excessive vibration. FADEC fault codes are standardized codes which are known in the industry to represent states of a system. Intermittent fault code tracking is a mechanism to track the existence of FADEC fault codes during the operation of a specific system. Thermal history measures and records the temperatures reached during operation of an engine.

[0036] Fig. 6 shows a fusion algorithm process map for the information fusion tool shown in Fig. 5. In Fig. 6 there is an architecture that includes eight layers (i.e., layers 0, 1, ... 7). The first layer is a concurrency layer 106, which is a pre-processing step to the main fusion operations. The concurrency layer 106 uses a temporal aggregation module 108 to perform aggregation of information over time and collect information before the main information fusion process is triggered. In addition, the concurrency layer 106 uses a decision fading module 110 to perform an information fading operation when certain temporal conditions are met. An association layer 112 exploits information about preferred misclassifications of a classification tool using a correlating module 114. A scaling layer 116 uses a reliability norming module 118 and relevance scaling module 120 to integrate a priori information regarding the performance of a classification tool. A strengthening layer 122 reinforces an opinion about a class when several classification tools indicate the same state of a class, such as the existence of a fault. The strengthening layer 122 uses a reinforcement module 124 to accomplish this process. A weakening layer 126 discounts information if several tools disagree on the classification state and it also discounts information if any classification tool indicates several classes at the same time, by use of a few tools discount module 128 and a many faults discount module 130.

[0037] An evidence updating layer 132 incorporates secondary evidential information, which is not generated by a classification tool, via an evidential reinforcement module 134. This type of information is used only to support a classification state, not to discount a classification state. A tie-breaking layer 136 deals with high ranking system states (e.g., faults) having similar confidence levels. It employs two strategies: one is to provide a higher weight value to a class that traditionally occurs more often (i.e., a frequency module 140); the other is to weigh a class state (e.g., a fault) higher if it is more critical to successful operation of the system (i.e., a criticality module 138). A backscaling layer 142 performs a transformation by using a backscaling module 144, which makes the fused information interpretable to an end user. US Patent Application Serial Number 09/704,476, filed on November 3, 2000, and entitled "Information Fusion Of Classifiers In Systems With Partial Redundant Information" provides more information on this fusion algorithm process map.

[0038] A description of how the algorithm performance assessment system 28 can be used to design an information fusion tool such as the one shown in Figs. 5 and 6 follows. To design this type of information fusion tool one should establish that each heuristic implemented in the layered architecture contributes to an improvement of the result. To that end, a designer can break down the development process into smaller steps and adapt a strategy of complete design cycles for each heuristic. That is, each step encompasses iterations of conception, implementation, and testing. This development process reduces the re-design necessary after completion of the whole fusion architecture, which at that stage would be more difficult because the influence of individual heuristics would be masked in the overall architecture. At the onset, it is not clear what effect the addition of new modules of the algorithm will have on the overall performance and whether the chosen implementation of that heuristic will move the error into the desired direction. Similarly, the parameters are not known beforehand and at least a coarse tuning is needed to be carried out for each heuristic.

[0039] The first step in using the algorithm performance assessment system 28 to evaluate the design of such an information fusion tool is to have the designer determine the problem space and select a subset from this space. In this example, assume that the designer selects 2 diagnostic tools, 1 evidential tool and 2 possible fault states from a problem space of 3 diagnostic tools, 5 evidential tools and 7 fault states with continuous-valued variables. Each one of the states in the subset is then limited to take on 3 possible confidence values. This amounts to 729 solutions. Even for a very constrained problem such as this one, 729 is a number too large for expert evaluation and hence will be reduced using the DoE. In this example, a half factorial DoE is preferable for providing a conceptual set of faults for the entire design space. In particular, the half factorial DoE cuts 729 solutions to 45 experiments. Table 1 shows the results of the 45 experiments and the results of expert evaluation.

Tool A Fault 1	Tool B Fault 1	Evidence Fault 1	Tool A Fault 2	Tool B Fault 2	Evidence Fault 2	Target 1	Target 2
0	0	0	0	0	0	0	0
1	0	0	0	0	1	0.6	0
0	1	0	0	0	1	0.3	0
1	1	0	0	0	0	0.9	0
0	0	1	0	0	1	0	0
1	0	1	0	0	0	0.8	0
0	1	1	0	0	0	0.4	0
1	1	1	0	0	1	1	0
0	0	0	1	0	1	0	0.1
1	0	0	1	0	0	0.5	0
0	1	0	1	0	0	0.3	0
1	1	0	1	0	1	0.9	0.1
0	0	1	1	0	0	0	0
1	0	1	1	0	1	0.5	0
0	1	1	1	0	1	0.3	0
1	1	1	1	0	0	1	0
0	0	0	0	1	1	0	0.3
1	0	0	0	1	0	0.5	0.2
0	1	0	0	1	0	0.2	0.1
1	1	0	0	1	1	0.8	0.2
0	0	1	0	1	0	0	0.2
1	0	1	0	1	1	0.5	0.1
0	1	1	0	1	1	0.3	0.2
1	1	1	0	1	0	0.9	0.1
0	0	0	1	1	0	0	0.6
1	0	0	1	1	1	0.4	0.5
0	1	0	1	1	1	0	0.4
1	1	0	1	1	0	0.6	0.4
0	0	1	1	1	1	0	0.7
1	0	1	1	1	0	0.5	0.5
0	1	1	1	1	0	0.2	0.5
1	1	1	1	1	1	0.8	0.5
0	0.5	0.5	0.5	0.5	0.5	0.3	0.3
1	0.5	0.5	0.5	0.5	0.5	0.7	0.3
0.5	0	0.5	0.5	0.5	0.5	0.3	0.3
0.5	1	0.5	0.5	0.5	0.5	0.5	0.2
0.5	0.5	0	0.5	0.5	0.5	0.4	0.3
0.5	0.5	1	0.5	0.5	0.5	0.4	0.3
0.5	0.5	0.5	0	0.5	0.5	0.4	0.2
0.5	0.5	0.5	1	0.5	0.5	0.4	0.4
0.5	0.5	0.5	0.5	0	0.5	0.4	0.3
0.5	0.5	0.5	0.5	1	0.5	0	0.4
0.5	0.5	0.5	0.5	0.5	0	0.4	0.3
0.5	0.5	0.5	0.5	0.5	1	0.4	0.3
0.5	0.5	0.5	0.5	0.5	0.5	0.4	0.3

Table 1: DoE set and expert specified target output

The columns labeled "Target 1" and "Target 2" contain the result of the expert evaluation, which represents the diagnosis for Fault 1 and Fault 2, respecrively. The DoE in this example is defined as a "bottom-up DoE" because the diagnostic expert is exposed only to the tool output without knowledte of the faults.

[0040] Next, the algorithm performance assessment system 28 evaluates the results of the experiments using a performance metric and a baseline algorithm. In this example, an index aggregating the weighted components of false positives, false negatives, and false classifieds, or any other suitable metric relative to a baseline algorithm can be used as the performance metric, however, other metrics such as sum squared error (SSE), total error rate (TER), total success rate (TSR), can be used. The baseline algorithm may comprise a benchmark algorithm that performed a maximum wins strategy on transformed and normalized input scaled by the a priori reliability of the diagnostic tool. In addition, an overall performance index that weighs the individual components false positives (FP), false negatives (FN), and false classified (FC) is used. The weightings of the individual components are driven by the following equation:

$$0.6*(1-FP)+0.3*(1-FN)+0.1*(1-FC)$$

In this example, the benchmark performance index is set to zero. Therefore, an increase in performance is measured as the fraction of improvement from that baseline to perfect performance, expressed in percent. The development process comprises the iterative addition of new algorithm components and refinement of existing components. In this example, the layers in the information fusion architecture roughly represent the final set of components. Each addition or change is then subjected to the evaluation process. Only components that improve the performance index are included in the overall architecture, thus allowing immediate evaluation. This iterative evaluation process continues until all the results for all of the components of the algorithm are acceptable.

[0041] Once all the results are acceptable the algorithm performance assessment system 28 simulates the actual system fault states ("top down") for the full input set with 3 diagnostic tools, 5 evidential tools, and 7 fault states to fine tune the algorithm and parameters. The Monte Carlo simulation allows each fault state to take on any value between 0 and 1 per architecture design requirements. The confusion matrices are used to create a system response that is indicative of realistic tool behavior. Table 2 shows an example of a confusion matrix that can be used in the simulation of the information fusion tool.

	$\hat{F}_0$	$\hat{F}_1$	$\hat{F}_2$	$\hat{F}_3$	$\hat{F}_4$	$\hat{F}_5$	$\hat{F}_6$
F_0	0.832	0.023	0.039	0.035	0.035	0.013	0.023
F_1	0.258	0.696	0.019	0.012	0.005	0.005	0.005
F_2	0.313	0.011	0.582	0.029	0.027	0.014	0.024
F_3	0.325	0.010	0.029	0.573	0.052	0.007	0.004
F_4	0.382	0.007	0.027	0.041	0.495	0.007	0.041
F_5	0.094	0.001	0.013	0.005	0.012	0.847	0.028
F_6	0.234	0.007	0.032	0.004	0.058	0.026	0.639

Table 2: Confusion Matrix

In this example, the confusion matrices of the diagnostic tools are the primary sources of a priori information for the information fusion tool. As a performance measure for the individual diagnostic tools, the confusion matrices list the observed faults versus the estimated faults. Because all faults are enumerated, it is possible to obtain information not only about the correctly classified states, but also about the false positives (FP), false negatives (FN), and false classified (FC) states. In Table 2, the rows list the actual faults, the columns list the estimated faults. Note that the fault F_0 represents the no fault condition. The diagonal entries of the confusion matrix represent the correctly classified cases. The first row, except the first entry, contains the FP state. The first column, except the first entry, contains the FN state. The off-diagonal elements, except the FP and FN states, are the FC states. Table 2 also shows the normalized confusion matrix for a diagnostic tool where each entry of a row was divided by the number of experiments for each class (i.e., the sum of the entries of the respective rows). The faults are denoted as F_n where n={0, ... , 6}.

[0042]

The Monte Carlo simulation in this example generally comprises three parts. One part includes computing the z-score, which as mentioned above, is a statistical measure indicative of how many standard deviations away from the mean the particular value is located, for a given a fault case. The z-score is computed by interpreting the associated entries in the confusion matrix as the area under the curve of a Gaussian distribution from which the z-score is obtained through iterative integration. Another part of the Monte Carlo simulation includes assigning random values based on the z-score. As an example, assume that the reliability of a diagnostic tool for a particular fault is 95%. This value is obtained from the diagonal entries of the confusion matrix, e.g., 0.95. The interpretation is that the diagnostic tool classifies correctly in 95% of the cases. Therefore, 95% of the output cases are generated between 0.5 and 1 and 5% of the output cases are generated between 0 and

0.5. Fig. 7 illustrates an example curve that can be used to simulate a diagnostic tool output. The third part of the simulation is to carry out the system run, in this case the information fusion. These parts of the Monte Carlo simulation are repeated many times until statistically meaningful results have been obtained. The results are then compiled in confusion matrix format.

[0043] After performing the simulation, the simulation performance component 44 determines if the results are acceptable. The results are acceptable if the performance index is greater than a predetermined threshold. If necessary, the logic or parameters and/or algorithm for a particular heuristic in the algorithm are adjusted. The process of determining if the simulation results are acceptable is repeated until a desired system response is obtained.

[0044] It is apparent that there has been provided in accordance with this invention, an algorithm performance assessment system, method and computer product. While the invention has been particularly shown and described in conjunction with a preferred embodiment thereof, it will be appreciated that variations and modifications can be effected by a person of ordinary skill in the art without departing from the scope of the invention.